[Introduction 3](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887799)

[What is C Language? 4](#_Toc171887800)

[What is low & height level language? 4](#_Toc171887801)

[What are Language Translators? 4](#_Toc171887802)

[Features of C language? 5](#_Toc171887803)

[What is Blueprint of coding in C? 5](#_Toc171887804)

[How C language code is executed? 6](#_Toc171887805)

[What is Boilerplate of C? 6](#_Toc171887806)

[Variable, Constant, Keyword 7](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887807)

[What is Variable? 8](#_Toc171887808)

[What is Data Type? 8](#_Toc171887809)

[Variable Rules? 9](#_Toc171887810)

[What is Constant? 9](#_Toc171887811)

[Types of Constant? 9](#_Toc171887812)

[Diff B/w Variable & Constant? 10](#_Toc171887813)

[What is Keyword? 10](#_Toc171887814)

[I/O, Comment 12](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887815)

[What is Comment & its types? 13](#_Toc171887816)

[Ways to take input from user? 13](#_Toc171887817)

[Instruction, Operator 15](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887818)

[What is instruction? 16](#_Toc171887819)

[What are operators and it’s types? 18](#_Toc171887820)

[Conditional Statement 22](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887821)

[What is Conditional Statement? 23](#_Toc171887822)

[What is “if” statement? 23](#_Toc171887823)

[What is “if-else” statement? 23](#_Toc171887824)

[What is “if-else if” statement? 24](#_Toc171887825)

[What is “Ternary” statement? 24](#_Toc171887826)

[What is “switch” statement 25](#_Toc171887827)

[Loop 26](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887828)

[What is Loop & it’s uses? 27](#_Toc171887829)

[What is “For” Loop? 27](#_Toc171887830)

[What is “While” Loop? 27](#_Toc171887831)

[What is “Do While” Loop? 27](#_Toc171887832)

[What is Break & Continue statement? 28](#_Toc171887833)

[What is Nested Loop? 28](#_Toc171887834)

[7 Function 29](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887835)

[What is function and it’s use? 30](#_Toc171887836)

[What are benefits of function? 30](#_Toc171887837)

[What is call by value in a function? 30](#_Toc171887838)

[Working of function? 30](#_Toc171887839)

[Type of function? 30](#_Toc171887840)

[Parts of function? 30](#_Toc171887841)

[Diff B/W Argument & Parameter? 30](#_Toc171887842)

[Function with diff number of Argument & Parameter? 31](#_Toc171887843)

[Function with Pointer? 31](#_Toc171887844)

[What is Recursion? 31](#_Toc171887845)

[Pointer 32](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887846)

[What is Pointer? 33](#_Toc171887847)

[How to assign pointer to a value? 33](#_Toc171887848)

[What is Dereferencing? 34](#_Toc171887849)

[How to assign Pointer to Pointer? 35](#_Toc171887850)

[What is Pointer Arithmetic? 35](#_Toc171887851)

[Increment & Decrement of Pointer? 35](#_Toc171887852)

[What is “Void” Pointer? 35](#_Toc171887853)

[What is “null” Pointer? 35](#_Toc171887854)

[What is Dangling Pointer? 35](#_Toc171887855)

[What is Wild Pointer? 35](#_Toc171887856)

[Role of Pointer in function? 35](#_Toc171887857)

[Array 36](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887858)

[What is Array? 37](#_Toc171887859)

[Initialization of Array? 37](#_Toc171887860)

[Input & Output in Array? 37](#_Toc171887861)

[Memory Representation of Array? 37](#_Toc171887862)

[What is Traverse an Array? 37](#_Toc171887863)

[Arrays as Functional Argument? 37](#_Toc171887864)

[What is Multidimensional Array? 37](#_Toc171887865)

[Memory Representation of 2D Array? 37](#_Toc171887866)

[Role of Pointer in Array? 37](#_Toc171887867)

[String 38](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887868)

[What is String? 39](#_Toc171887869)

[Initializing a String? 39](#_Toc171887870)

[What Happens in Memory? 40](#_Toc171887871)

[String Format Specifier? 40](#_Toc171887872)

[String I/O? 40](#_Toc171887873)

[String Functions? 41](#_Toc171887874)

[Role of Pointer in String? 41](#_Toc171887875)

[Structures 42](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887876)

[What is Structures? 43](#_Toc171887877)

[Syntax of Structures? 43](#_Toc171887878)

[Memory allocation for Structures? 43](#_Toc171887879)

[Benefits of using Structures? 43](#_Toc171887880)

[Array of Structures? 43](#_Toc171887881)

[Initializing of Structures? 43](#_Toc171887882)

[Pointers in Structures? 43](#_Toc171887883)

[What is Arrow Operator & it’s use? 43](#_Toc171887884)

[Passing Structure to Function? 43](#_Toc171887885)

[What is Typedef keyword? 43](#_Toc171887886)

[File I/O 44](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887887)

[What is File I/O? 45](#_Toc171887888)

[Operation for Files & Types of Files? 45](#_Toc171887889)

[Role of Pointer in File Handling? 45](#_Toc171887890)

[Opening & Closing of a File? 45](#_Toc171887891)

[File opening Modes? 45](#_Toc171887892)

[Reading & Writing of a file? 45](#_Toc171887893)

[Read & Write a Character? 45](#_Toc171887894)

[What is End of File? 45](#_Toc171887895)

[Dynamic Memory Allocation 46](file:///D:\B.Tech\%5b%20All%20projects%20%5d\2-%20Git%20not%20posted\Notebook\BCA\C\C%20.docx#_Toc171887896)

[What is Dynamic Memory Allocation? 47](#_Toc171887897)

[Function for DMA? 47](#_Toc171887898)

[Extra Commands 48](#_Toc171887899)

1

Introduction

**Objectives**

* What is C Language?
* What is low & height level language?
* What are Language Translators?
* Features of C language?
* What is Blueprint of coding in C?
* How C language code is executed?
* What is Boilerplate of C?

What is C Language?

It is written by Dennis Ritchie in 1972 at Bell Labs. It is successor of B language. It is standardized by ANSI in 1989.

It is also called mother of all language.

What is low & height level language?

|  |  |
| --- | --- |
| LLL | HLL |
| It includes Machine & Assembly language. | It includes C, C++, Java etc. |
| Machine Language: Include code written in 0 & 1.  It is completely CPU dependent or we can say machine dependent. So, for it we need to know about system  architecture.  But execution is fast as it not needed further compilation.  It is hard to understand.  It not portable. | It is very close to human language so it is easy to  understand. |
| Assembly Language: Include symbols as for + we can use Add.  It is little bit understandable as compare to machine language.  It slows as compare to machine language as it need further compilation.  It is also not portable. | It is slow as it need to compile before execution. |

What are Language Translators?

There are 3 types of Language Translators:

1. Assembler
2. Compiler
3. Interpreter

Assembler – It converts code written in assembly language to machine language. It Is for LLL.

Compiler & Interpreter both are for HLL.

|  |  |
| --- | --- |
| **Compiler** | **Interpreter** |
| HLL 🡪 compiler 🡪 Machine Language  It converts code before execution. | HLL 🡪 compiler 🡪 Machine Language  It converts code while execution. |
| It takes whole code and convert it into object code.  Here it converts code into object code before running so here execution and translator not run parallelly. | It takes code line by line and give it output.  Means it will convert code into object code while the code is running.  Means execution and translator run parallelly. |
| Here separate file of object code is generated. | Here no separate file of object code is generated. |
| It take less time. | It take more to time. |
| It will give list errors in last. | It stop on the line where it gets the error so it is helpful in debugging of code. |
| Here once the object file is generated we can run that file directly means we not need main code file but that object file is platform dependent. | Here no separate object file is created so we need main code file each time. |
| As it generates a file for object code so it need more space. | As it not generates a file for object code so it need less space. |
| Source code 🡪 compiler 🡪 object code / file 🡪 output | Source code 🡪 interpreter 🡪 output |

If a Russian speaking guy meets a Hindi speaking guy, then he needs a translator who will translate line by line for him, in this case he needs a translator each time he wants to talk to a Russian guy. **[Eg of Interpreter]**

If he gets a Russian movie which is with Eng sub he not need translator each time he wants to watch that movie he only need translation. **[Eg of compiler]**

**Note :**

1. The code is compiled by the CPU & machine code can be different for different CPU as it depends on the architecture of CPU. Architecture of CPU Eg is some computers are 32-bit & some are 64-bit.
2. Machine code is called object code.

**Note :**

1. The code is compiled by the CPU & machine code can be different for different CPU as it depends on the architecture of CPU. Architecture of CPU Eg is some computers are 32-bit & some are 64-bit.
2. Machine code is called object code.

Features of C language?

1. It is HLL.
2. Small language having 32 keywords.
3. It is system independent in maximum cases.
4. It has Built in Functions & operators.
5. It is Structure Language / Modular.
   1. **Structure Language means we can arrange code in 1or more functions.**
   2. **Modular means we can break code in diff functions.**
6. It has Pointers.
7. It has dynamic memory.
8. It is extensible Language.
   1. **extensible Language means we can create our own function.**
9. Core of many programing language as many programming languages core is written in C language.
10. Its compilation & execution is fast.
11. It is case sensitive.
12. It is use as embedded language.
    1. **embedded language means it is use to make functioning of products like Ref, washing machine, TV etc.**

What is Blueprint of coding in C?

1. Documentation Section

It is included Author Name, Date, Program is about what

1. Link Section

It includes the link file like stdio, conio etc.

1. Definition Section
2. Global Declaration
3. Main Section
4. Sub Program Section

How C language code is executed?

What is Boilerplate of C?

Boiler plate of C

**# include<stdio.h>**

**int main(){**

**// Code**

**return 0;**

**}**

2

Variable, Constant, Keyword

2

**Var****iable, Constant, Keyword**

* What is Variable?
* What is Data Type?
* Variable Rules?
* What is Constant?
* Constant Types?
* Diff B/w Variable & Constant?
* What is Keyword?

**Objectives**

**Objectives**

What is Variable?

Variable is a name given to memory where we can store any type of value.

In C we also write the type of data before variable name.

|  |  |
| --- | --- |
| **Variable Prefix** | **Data Type** |
| Int | Integer |
| float | Decimal |
| char | Character |

It is also called **Identifier**.

Doubt why void a = 45; Wrong

Code Format:

int number = 45;

float percentage = 42.8;

char star = '\*';

According to ASCI length of variable name is 31 Character, but that doesn’t mean that variable name can’t be greater than 31 Character but that means that all the compile will consider variable name unique up to 31 Character (Minimum) but it can vary according the compiler. As some compiler can consider variable name unique up to 200 Character but there will not be a compile which consider variable name unique below 31 Character.

What is Data Type?

It tell the type of Data stored in variable and how memory is allocated to that variable.

|  |  |
| --- | --- |
| **Data Type** | **Size in Bytes** |
| Char or signed char | 1 |
| Unsigned char | 1 |
| int or signed int | 2 |
| Unsigned int | 2 |
| Short int or Unsigned short int | 2 |
| Signed short int | 2 |
| Long int or Signed long int | 4 |
| Unsigned long int | 4 |
| Float | 4 |
| Double | 8 |
| Long Double | 10 |

There are sub categories ofint, char, float, double Which is short & long , sign & unsign. Difference in them is that they have different range and diff memory.

Benefit of using data type according to requirement is that it saves memory as we need to store integer value but we use float data type for it, in this case as int need 2 bytes of memory & float need 4 bytes it will not effect in small level but at bigger where Lakhs or billion of data is generate through that variable there it make a very big difference.

**Note :** Range and Memory of data type depend on system.

**Note :** Range and Memory of data type depend on system.

Variable Rules?

There are some rules to name a variable.

* + - Variables are case sensitive.
    - 1st character is alphabet or '\_'.
    - no comma/blank space.
    - No other symbol other than '\_'.

What is Constant?

**Constants** are those value which not change with time. If the value changes will code is running or enter while code is going to run then it is called **variable**.

Types of Constant?

We have 1 more type of Constant which is called Symbolic Constant. It is written outside the “main” section.

**#include <stdio.h>**

**#define pi 3.14**

**int main()**

**{**

**float ans = pi;**

**printf("value of ans %f", ans);**

**return 0;**

**}**

Here **#define pi 3.14** is the constant or more specific Symbolic Constant, it can be assign to another variable as in **above code** or can be directly use as in **below code**.

**#include <stdio.h>**

**#define pi 3.14**

**int main()**

**{**

**printf("value of pi %f", pi);**

**return 0;**

**}**

Also Write constant before a variable name to make it constant.

Diff B/w Variable & Constant?

|  |  |
| --- | --- |
| **Constants** | **Variable** |
| **Constants** are those value which not change with time. | If the value changes will code is running or enter while code is going to run then it is called **variable**. |

|  |  |
| --- | --- |
| **Constants** | **Variable** |
| **Constants** are those value which not change with time. | If the value changes will code is running or enter while code is going to run then it is called **variable**. |

What is Keyword?

Keywords are those which are reserved by language means it can’t be use as name of a variable.

In C keywords are in lowercase only, so if we write any keyword in uppercase then it is not considered as keyword.

Eg : **int – It is a keyword**

**Int – It is not a keyword**

Eg : **int – It is a keyword**

**Int – It is not a keyword**

C has 32 Keywords only.

|  |  |  |  |
| --- | --- | --- | --- |
| **auto** | **double** | **int** | **struct** |
| **break** | **else** | **long** | **switch** |
| **case** | **enum** | **register** | **typedef** |
| **char** | **extern** | **return** | **union** |
| **continue** | **for** | **signed** | **void** |
| **do** | **if** | **static** | **while** |
| **default** | **goto** | **sizeof** | **volatile** |
| **const** | **float** | **short** | **unsigned** |

3

I/O, Comment

3

**I/O, Comment**

* What is Comment & its types?
* Ways to take input from user?
* Ways to give output?

**Objectives**

**Objectives**

What is Comment & its types?

Comments are text which compiler ignore. It is use to write some kind of information about code or some other thing.

There are 2 types of comment :-

* Single Line Comment
  + It is written in “//”
* Double Line Comment
  + It is written in “/\* \*/”

Ways to take input from user?

To take input from user we will use **“scanf”.**

scanf("%variable\_type ",&variabl\_name);

|  |  |
| --- | --- |
| **Variable Type** | **Variable Name** |
| %d | Integer |
| %f | Real |
| %c | Character |

‘&’ sign must come before variable name.

Scanf store value which will be entered in variable we give it but type of data enter and variable must be same.

Eg : scanf(“%d”,&age);

If enter **18.00** in it, it will show error as **18.00** is **real type** and for real type %f is used.

But if we enter **18** in it, then code will run properly as “**18**” is **integer type**.

Eg : scanf(“%d”,&age);

If enter **18.00** in it, it will show error as **18.00** is **real type** and for real type %f is used.

But if we enter **18** in it, then code will run properly as “**18**” is **integer type**.

**Ways to give output?**

**Note :** We can’t write a string in scanf as if do so then this will happen.

# include<stdio.h>

int main(){

    int age ;

    scanf("What is your age %d",&age);

    printf("Age is %d",age);

    return 0;

}

Output: 45

Age is 0

**Note :** We can’t write a string in scanf as if do so then this will happen.

# include<stdio.h>

int main(){

    int age ;

    scanf("What is your age %d",&age);

    printf("Age is %d",age);

    return 0;

}

Output: 45

Age is 0

So we can only write like this

# include<stdio.h>

int main(){

    int age ;

    scanf("What is your age %d",&age);

    printf("%d",age);

    return 0;

}

Output: 45

Age is 45

If we want a text to display before entering a value we can write like this.

# include<stdio.h>

int main(){

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    printf("Age is %d",age);

    return 0;

}

 Output:    What is your age 45

            Age is 4

So we can only write like this

# include<stdio.h>

int main(){

    int age ;

    scanf("What is your age %d",&age);

    printf("%d",age);

    return 0;

}

Output: 45

Age is 45

If we want a text to display before entering a value we can write like this.

# include<stdio.h>

int main(){

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    printf("Age is %d",age);

    return 0;

}

 Output:    What is your age 45

            Age is 4

4

Instruction, Operator

4

**Instruction, Operator**

* What is instruction?
* What are operators and it’s types

**Objectives**

**Objectives**

What is instruction?

Type of declaration

|  |  |
| --- | --- |
| Right | Wrong |
| Int a=45, b=8; |  |
| Int a=4+5-7/2\*5; |  |
| Variable must be defined before using it | Int b=a;  Int a=5; |
| Int a,b,c;  A=b=c=5; | Int a=b=c=5; |

**Note :**

1. C allows only one variable on left-hand side of =. That is, z = k \* l is legal, whereas k \* l = z is illegal.
2. The modulus operator (%) cannot be used with floats. Also note that on using %, sign of the remainder is always same as the sign of the numerator. Thus -5 % 2 yields -1, whereas, 5 % -2 yields 1.
3. Each operator must be written.

a = c.d.b(xy) //Right for maths & Wrong for coding

a = c \* d \* b \* ( x \* y ) //Right for maths & Right for coding

1. To do exponential we need to use pow()

But it is only work for float type.

Eg :

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Result** | **Operation** | **Result** |
| 5/2 | 2 | 2/5 | 0 |
| 5.0/2 | 2.5 | 2.0/5 | 0.4 |
| 5/2.0 | 2.5 | 2/.5.0 | 0.4 |
| 5.0/2.0 | 2.5 | 2.0/5.0 | 0.4 |

Doubt Does Forceful conversion is possible

|  |  |
| --- | --- |
| **Operation** | **Result** |
| Int + int | Int |
| real + real | Real |
| Integer + Real | Real |

It is very important

|  |  |  |
| --- | --- | --- |
| Precedence Table | | |
| Description | **Operator** | **Associativity** |
| Postfix increment / decrement Function call  Array subscripting  Structure / Union member access  Structure / Union member access | **++ --**  **( )**  **[ ]**  **.**  **->** | **Left to Right** |
| Prefix increment / decrement  Unary plus and minus  Logical not, Bitwise not  Typecast Value of address (dereference)  Address of  Size in bytes | **++ --**  **+ -**  **! ~**  **( type )**  **\***  **&**  **sizeof** | **Right to left** |
| Multiplication, Division, Modulus | **\* / %** | **Left to Right** |
| Addition, Subtraction | **+ -** | **Left to Right** |
| Bitwise left shift, Bitwise right shift | **<< >>** | **Left to Right** |
| Less than, Less than or equal to  Greater than, Greater than or equal to | **< <=**  **> >=** | **Left to Right** |
| Equal to, Not equal to | **== !=** | **Left to Right** |
| Bitwise AND | **&** | **Left to Right** |
| Bitwise XOR | **^** | **Left to Right** |
| Bitwise OR | **|** | **Left to Right** |
| Logical AND | **&&** | **Left to Right** |
| Logical OR | **||** | **Left to Right** |
| Conditional | **? :** | **Left to Right** |
| Assignment | **=**  **+= -=**  **\*= /= %=**  **<<= >>=**  **&= ^= |=** | **Right to left** |
| Comma | **,** | **Left to Right** |

When we need to decide priority of same order means as **\* / %** has same level of priority in this priority is from left to right means if expression is **int** **a = 3 % 2 \* 5 / 2;**  as all operator in this expression has same level of priority so it will be evaluated from left to right.

**3 % 2 \* 5 / 2**

**1 \* 5 / 2**

**5 / 2**

**2**

Eg : i = 2 \* 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

Stepwise evaluation of this expression is shown below:

i = 2 \* 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8 operation: **\***

i = 1 + 4 / 4 + 8 - 2 + 5 / 8 operation: **/**

i = 1 + 1+ 8 - 2 + 5 / 8 operation: **/**

i = 1 + 1 + 8 - 2 + 0 operation: **/**

i = 2 + 8 - 2 + 0 operation: **+**

i = 10 - 2 + 0 operation: **+**

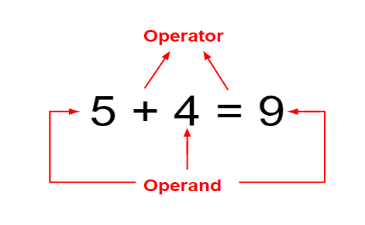
i = 8 + 0 operation : **-**

i = 8 operation: **+**

What are operators and it’s types?

Operators indicate what action or operation to perform.

Operands indicate what items to apply the action to.



Eg :

Type of operator based on number operands

1. Unary - When operation is done on 1 operand.
2. Binary - When operation is done on 2 operands.
3. Ternary - When operation is done on 3 operands.

Unary include those all case when operation is done one operator.

1. Unary Minus (-)

It allows us to use negative value.

**#include <stdio.h>**

**int main()**

**{**

**int num = -55;**

**printf("Value of num is %d \n",num);**

**return 0;**

**}**

Output: Value of num is -55

1. Increment (++), Decrement (--)

|  |  |  |  |
| --- | --- | --- | --- |
| **i++** | In this case value of i is used first than it increases | **++i** | In this case value of i is first increase than it is used |
| **i--** | In this case value of i is used first than it decreases | **--i** | In this case value of i is first decrease than it is used |

#include <stdio.h>

int main()

{

    for(int i=1; i<=5; i++){

        printf("Value of 'i' is %d \n",i);

    }

    return 0;

}

Output: Value of 'i' is 1

Value of 'i' is 2

Value of 'i' is 3

Value of 'i' is 4

Value of 'i' is 5

1. Logical Operator (!)

It allows us to convert true into false and vice versa.

#include <stdio.h>

int main()

{

    if(!(10>15)){

        printf("You statement is wrong");

    }

    return 0;

}

Output: You statement is wrong

As according to logic “if” statement must nit be executed as (10>15) will return false but due to “!” false turn into true so it is executed.

1. Address of (&)

Used in Pointer.

1. sizeof

It allows us to find what size is take by any keyword.

#include <stdio.h>

int main()

{

    char star = '\*';

    printf("Size of %d \n",sizeof(star));

    printf("Size of %d \n",sizeof(int));

    return 0;

}

Output: Size of 1

Size of 4

Binary includes those case where 2 operands are present.

1. Arithmetical Operator
2. Assignment Operator
3. Logical Operator
4. Bit wise Operator
5. Comma Operator
6. Relational Operator
7. Equality Operator

Ternary includes those case where 3 operands are present.

It has one 1 eg.

Syntax: **condition? True statement: False statement;**

#include <stdio.h>

int main() {

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    age>18? printf("He is an adult"):printf("He is not an adult");

    return 0;

}

Output: What is your age **15**

**He is not an adult**

Type of operators are:

1. Arithmetical Operator
2. Assignment Operator
3. Logical Operator
4. Bit wise Operator
5. Comma Operator
6. Relational Operator
7. Equality Operator
8. Precedence & Associativity
9. Arithmetical Operator (+, -, \*, /, %)

|  |  |
| --- | --- |
| + | Addition |
| - | Substation |
| \* | Multiply |
| / | Divide – It gives Quotient |
| % | Modulus – It give Remainder |

Rules for Modulus – It can only be used by integer values.

It takes sign of 1st value from left side.

1. Assignment Operator (=)

It is used to assign a value to a Variable.

Rule 🡪 It’s LHS Must have single variable.

Means **a+b = c (Wrong)** **c = a+b (Right)**

Value is assigned from Right to Left

Means a = b = c = d = 10;

In this case first “d” is assigned 10 then to “c” then to” b” then to “a”.

|  |  |
| --- | --- |
| Original Expression | Shorthand |
| c=c+b | c += b |
| c=c-b | c -= b |
| c=c\*b | c \*= b |
| c=c/b | c /= b |

1. Logical Operator

|  |  |
| --- | --- |
| && | 1. It returns true only if all conditions are true 2. It ends the check when single condition becomes false |
| || | 1. It returns false even if one condition is false 2. It checks all the conditions |
| ! | It converts true into false and vice versa |

**///////////////Left /////////////**

1. Bit wise Operator

It is use to perform operation on bit level means the smallest level.

But it only be performed on integer and character values.

1. Comma Operator
2. Relational Operator

It return True & False

|  |  |
| --- | --- |
| < | Greater than |
| > | Less than |
| <= | Greater than or equal to |
| >= | Less than or equal to |
| == | Equal to |
| != | Not Equal to |

**Note :** We can compare character with it but we can’t compare string with it as in that case it compares address value first character which we don’t know so we can get unexpected output.

**Note :** We can compare character with it but we can’t compare string with it as in that case it compares address value first character which we don’t know so we can get unexpected output.

1. Equality Operator
2. Precedence & Associativity

5

Conditional Statement

5

**Conditional Statement**

* What is Conditional Statement?
* What is “if” statement?
* What is “if-else” statement?
* What is “if-else if” statement?
* What is “Ternary” statement?
* What is “switch” statement

**Objectives**

**Objectives**

What is Conditional Statement?

Conditional statement are those statements that allow a program to execute different blocks of code based on whether a certain condition is true or false.

In C, we have many conditional statements like if, if else, if else-if.

What is “if” statement?

It allows us to execute a code block only when a certain condition is meant.

# include<stdio.h>

int main(){

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    if(age >= 18){

        printf("You are adult");

    }

    return 0;

}

Output: What is your age **45**

**You are adult**

What is “if-else” statement?

It allow us to execute different block code according to a situation is true or false. It has 2 parts if and else, “if” part contains condition according to which we want to execute a code block and has the code which has to executed if condition is full filled means it is true & “else” part contains that block of code which has to be executed is condition is not full filled means it is false.

# include<stdio.h>

int main(){

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    if(age >= 18){

        printf("You are adult");

    }

    else{

        printf("You are not adult");

    }

    return 0;

}

Output: What is your age **45**

**You are adult**

What is “if-else if” statement?

It allow us to execute different block code according to a situation is true or false. It has 3 parts if, elseif, else. If & else part is same as if else and “elseif” part allow us to add more and more condition in it.

# include<stdio.h>

int main(){

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    if(age >= 18){

        printf("You are adult");

    }

else if(12<=age && 18>=age){

        printf("You are Tenager");

    }

    else{

        printf("You are not adult");

    }

    return 0;

}

Output: What is your age **15**

**You are Teenager**

What is “Ternary” statement?

It is shorthand of if else Conditional statement.

Syntax: **condition? True statement: False statement;**

#include <stdio.h>

int main() {

    int age ;

    printf("What is your age ");

    scanf("%d",&age);

    age>18? printf("He is an adult"):printf("He is not an adult");

    return 0;

}

Output: What is your age **15**

**He is not an adult**

What is “switch” statement

Switch Statement allows us to make multi conditional code block.

Syntax :

swtch(variable\_value){

Case XYZ:

// Code

break;

Case ABC:

// Code

break;

Default:

// Code

break;

}

XYZ & ABC represent the value to which we have to match with variable value and execute the code but “\*break” statement is required to exit the switch condition statement block otherwise code of all the below the case whose value matches with variable value will also executed.

**Note :** In C language

1. Cases can be in any order, means case name not need to in alphabetical or numerical order.
2. Nested switch are allowed.
3. Name of switch case can’t be a string.

Doubt File Related to Switch Case [File](Doubt%20Files/Switch%20Statement.md)

6

Loop

* What is Loop & it’s uses?
* What is “For” Loop?
* What is “While” Loop?
* What is “Do While” Loop?
* What is Break & Continue statement?
* What is Nested Loop?

**Objectives**

What is Loop & it’s uses?

It helps to repeat some parts of the program.

In C, we have 3 types of Loop.

* For
* While
* Do while

What is “For” Loop?

This type of loop not have any special thing .

Syntax :

**for(initialisation; condition; update){**

**//Code**

**}**

In this if we not give update condition then it will become infinite loop.

What is “While” Loop?

This type of loop not have any special thing.

Syntax :

**int counter = 1; // initialisation**

**While(counter > 0) // condition**

**//Code**

**Counter++; // update condition**

**}**

In this if we not give update condition then it will become infinite loop.

What is “Do While” Loop?

This type of loop will execute minimum 1 time.

Syntax :

**int counter = 1; // initialisation**

**do{**

**//Code**

**counter++; // update condition**

**} While(counter > 0) // condition**

In this if we not give update condition then it will become infinite loop.

What is Break & Continue statement?

Break Statement 🡪 It allows us to exit a loop, but it only work if it is inserted in a loop otherwise it will throw an error.

Continue Statement 🡪 It allow us jump or skip a step.

**# include<stdio.h>**

**int main(){**

**for(int i=1; i<=10; i++){**

**if(i == 5){**

**continue;**

**}**

**if(i == 9){**

**break;**

**}**

**printf("Value is %d \n",i);**

**}**

**return 0;**

**}**

**Output: Value is 1**

**Value is 2**

**Value is 3**

**Value is 4**

**Value is 6**

**Value is 7**

**Value is 8**

Due to “continue” 5 is not printed & due to “break” loop ends at 9 but 9 is not printed as printing statement is after break condition but loop is exited at break statement so 9 is also not printed.

What is Nested Loop?

**///////Left//////////**

When we write a loop inside a loop it is called nested loop.

7 Function

* What is function and it’s use?
* Diff Types of Syntax of function?
* Properties & Types of Function?
* Diff B/W Argument & Parameter?
* Function with pointer?
* What is Recursion?

**Objectives**

Is main function is user defined or not also tell reason for your ans?

What is function and it’s use?

Function allows us to reuse a block code for a particular task.

It is advised to use meaningful name for function.

What are benefits of function?

* It allows to use a block of code again & again.
* It also helps in debugging as we now where to look and upto where to look.

Working of function?

When interpreter reach the function calling line, it transfer the control to the function and when code of function is executed completely the control comes back to main function, & the memory space taken by the variable of function is freed.

Type of function?

1. **Static**

In these, function code is added at the time of compilation at the place of its calling.

1. **Dynamic**

In these, the control is shifted to function code at the time of calling and when the function execution is completed it shifts to main function and memory of function is cleared.

Parts of function?

1. Function Declaration/Prototype
2. Function Definition
3. Function Calling

**return\_type function\_name(parameter){   // Function Prototype**

**// code                             // Function Defination**

**}**

**function\_name(argument)                 // Function Calling**

what is difference between (int, int) & (int a, int b)

At time of declaration we can write only data type (int, int) but at the time of definition we have to write data type + variable name (int a, int b).

Diff B/W Argument & Parameter?

|  |  |
| --- | --- |
| **Argument** | **Parameter** |
| Values that are passed in function call | Values in function declaration & Definition |
| Used to send value to function | Used to receive value by function |
| It is called Actual Parameter | It is called Formal Parameter |

**Note :** Sometimes if the definition & declaration of function has no parameters but call has arguments then compiler get confuse and generate output like

**#include<stdio.h>**

**void sum();**

**int main(){**

**sum(4,5);**

**return 0;**

**}**

**void sum(){**

**int a,b;**

**printf("Enter a & b \n");**

**scanf("%d \n %d",&a,&b);**

**printf("\nSum of %d & %d is %d", a,b,a+b);**

**}**

**Output: Enter a & b**

**6**

**4**

**Sum of 6 & 4 is 10**

So to prevent this we can use void in definition & declaration of function parameters like

**#include<stdio.h>**

**void sum(void);**

**int main(){**

**sum(4,5);**

**return 0;**

**}**

**void sum(void){**

**int a,b,sum=0;**

**printf("Enter a & b \n");**

**scanf("%d \n %d",&a,&b);**

**printf("\nSum of %d & %d is %d", a,b,a+b);**

**}**

**Note :** As in this code due to void compiler give error

**Function.c: In function 'main':**

**Function.c:5:5: error: too many arguments to function 'sum'**

**sum(4,5);**

**^~~**

**Function.c:2:6: note: declared here**

**void sum(void);**

**^~~**

So, if you want to prevent this situation use void.

Function with diff number of Argument & Parameter?

As if we use same name for different function with diff number f parameters than it is called function overloading, but C not support function overloading.

What is call by value & call by Reference in a function?

Call by value

// code of function with name – sum

Sum(arguments);

Means normal calling by name of functions with arguments if needed.

Call by reference

// code of function with name – sum

Sum(arguments);

But in the arguments are the address of the variable.

**Use**

**Call by value** 🡪 It is use when just need to call a function.

**Call by reference** 🡪 It is used when we need to call a function & also want than the changes happened to the variable inside the function can be use in the main function.

Code to know it practically

**#include<stdio.h>**

**void print(int a, int b){**

**// call by value**

**int c;**

**c=a;**

**a=b;**

**b=c;**

**printf("Inside the function\n");**

**printf("a=%d & b=%d \n",a,b);**

**}**

**int main(){**

**int a=5;**

**int b=10;**

**print(a,b);**

**printf("Outside the function\n");**

**printf("a=%d & b=%d \n",a,b);**

**}**

**Ouput: Inside the function**

**a=10 & b=5**

**Outside the function**

**a=5 & b=10**

**#include<stdio.h>**

**void print(int \*a, int \*b){**

**// call by reference**

**int c;**

**c=\*a;**

**\*a=\*b;**

**\*b=c;**

**printf("Inside the function\n");**

**printf("a=%d & b=%d \n",\*a,\*b);**

**}**

**int main(){**

**int a=5;**

**int b=10;**

**print(&a,&b);**

**printf("Outside the function\n");**

**printf("a=%d & b=%d \n",a,b);**

**}**

**Output: Inside the function**

**a=10 & b=5**

**Outside the function**

**a=10 & b=5**

Function with Pointer?

What is Recursion?

8

**Pointer**

* What is Pointer?
* How to assign pointer to a value?
* How to assign Pointer to Pointer?
* What is Pointer Arithmetic?
* Increment & Decrement of Pointer?
* What is “Void” Pointer?
* What is “null” Pointer?
* What is Dangling Pointer?
* What is Wild Pointer?
* Role of Pointer in function?

**Objectives**

What is Pointer?

A pointer is a variable that stores the memory address of another variable. Pointers provide a way to access and manipulate data stored in memory.

Syntax: Data\_Type \* Pointer\_Name = &Variable\_name;

How to assign pointer to a value?

Syntax: Data\_Type \* Pointer\_Name = &Variable\_name;

**#include <stdio.h>**

**int main() {**

**int var = 10;   // A normal integer variable**

**int \*ptr;      // A pointer to an integer**

**ptr = &var;    // Assigning the address of var to the pointer**

**printf("Value of var: %d\n", var);**

**printf("Address of var: %p\n", (void\*)&var);**

**printf("Value of ptr (address of var): %p\n", (void\*)ptr);**

**printf("Value pointed to by ptr: %d\n", \*ptr);**

**return 0;**

**}**

**Output: Value of var: 10**

**Address of var: 0061FF18**

**Value of ptr (address of var): 0061FF18**

**Value pointed to by ptr: 10**

As in above code we can see that we access the age value by pointer but for it we need to use the **%d format specifier** .

In we have seen **(void \*)** can point to any data type. By casting the pointer to void \*, you ensure that the pointer is treated correctly by “printf”. It has same work as %p but using **(void \*)**  is consider a good practice.

Doubt what is **(void \*)**  ask it from a teacher also?

**#include <stdio.h>**

**int main(){**

**int age = 55;**

**int \*p = &age;**

**printf("age is %d \n",age);**

**int age2 = \*p; // Value assign by Pointer**

**printf("age2 is %d \n",age2);**

**age2 = 18;**

**printf("age2 is %d \n",age2);**

**printf("age is %d",age);**

**return 0;**

**}**

**Output: age is 55**

**age2 is 55**

**age2 is 18**

**age is 55**

In this code we can see in detail how we can excess value by using pointer. But with this the original value will not change, means a copy original value is store not the original value.

**// We can make auto change value with help of pointer**

**# include <stdio.h>**

**int main(){**

**int age = 55;**

**int \*age2 = &age;**

**printf("age is %d",age);**

**printf("age2 value is %d \n",\*age2);**

**\*age2 = 45;**

**printf("age is %d",age);**

**return 0;**

**}**

**Output: age is 55**

**age2 value is 55**

**age is 45**

With this code we can use original value at more than 1 location without making it’s copy. So, if we made change in value at 1 location it will also change value at original location.

As in it we not made changes in age but in pointer (age2) than also value of age is changed.

What is Dereferencing?

As in above code in some eg we not use pointer not to access the address of variable but to access variable value and this process when we not access the address but the value of variable by pointer that is called dereferencing.

**# include <stdio.h>**

**int main(){**

**int age = 55;**

**int \*p = &age;**

**printf("Pointer value is %d \n",\*p);  // Dereferencing**

**return 0;**

**}**

**Output: Pointer value is 55**

How to assign Pointer to Pointer?

nlnl

What is Pointer Arithmetic?

nlnl

Increment & Decrement of Pointer?

nlnl

What is “Void” Pointer?

nlnl

What is “null” Pointer?

nlnl

What is Dangling Pointer?

nlnl

What is Wild Pointer?

nlnl

Role of Pointer in function?

nlnl

9

Array

**Objectives**

* What is Array?
* Initialization of Array?
* Input & Output in Array?
* What is Traverse an Array?
* Arrays as Functional Argument?
* What is Multidimensional Array?
* Memory Representation of Array?

What is Array?

Array is used to store multiple values in single variable .

Syntax : data\_type variable\_name[ array\_size ] = { array\_data };

Eg: int marks[5] = {5,4,6,8,2};

**Note :**

1. Array memory is continuous means if the first value of array is stored at address 1020 and size of array is 20 byte then in memory range of memory store of array is from 1020 to 1040.
2. To calculate the size of array we can use this formula 🡪 size of one element of array \* Total no. of elements of array
3. Address of array is equal to address of base element / first element of array.
4. Element of array must be of same type means if we put integer type in it then all elements should be integer.
5. int a[ ]; // Wrong int a [ ]={4,5,7}; // Right

means if we not define the size of array at the time of initialization then we must give the elements of array at the time of initialization but in this we can give any number of elements in this array.

1. We can also define the size of array in arithmetical manner means **int marks[2+2];**
2. We can’t use variable name in place of size but we can use macro in it’s place and use of macro is also preferred as it allows us to define the size of array at the time of running.

Initialization of Array?

Input & Output in Array?

Memory Representation of Array?

What is Traverse an Array?

Arrays as Functional Argument?

What is Multidimensional Array?

Memory Representation of 2D Array?

Role of Pointer in Array?

10

String

* What is String?
* Initializing a String?
* What Happens in Memory?
* String Format Specifier?
* String Functions?
* Role of Pointer in String?
* Standard Library Function for String?

**Objectives**

What is String?

Group of character make a string.

A string is 1-D array of character terminated by a null (‘\0’).

**Note :** Null (‘\0’) &’0’ are different as ASCII value of null (‘\0’) is 0 & ASCII value of ‘0’ is 48.

To terminating null (‘\0’) is important, because it is the only way the functions that work with a string can know where the string ends. In fact, a string not terminated by a ‘\0’ is not really a string, but merely a collection of characters.

Initializing a String?

To write a string we can declare it as shown below

**char str[40] = {’V’,’A’ ,’N’ ,’S’ ,’H’ ,’I’ ,’T’ ,’ ’ ,’G’ ,’A’ ,R’’ ,’G’,’\0’};**

So, Syntax is **char Variable\_name[Size\_of\_string] = {‘characte1’, ‘characte2’, …};**

But in this we need to use ‘\0’ to tell the compiler that it is not an array of character but a string, as string has it own roles and methods like “strlen()”.

**#include<stdio.h>**

**int main(){**

**char str[40] = {'V','A','N','S','H','I','T',' ','G','A','R','G','\0'};**

**int i=0;**

**while(str[i] != '\0'){**

**printf("%c",str[i]);**

**i++;**

**}**

**//printf("String is %c",str);**

**return 0;**

**}**

We can't print string just using **printf("String is %c",str);**  we need to loop it till the length of string or till we get ‘\0’.

To declare string without ‘\0’ we need to write in below manner

**char str[ ] = “VANSHIT GARG”;**

Syntax **char Variable\_name[ ] = “String”;**

**Note :** In above example some has empty square brackets “[ ]” and some has some value b/w square brackets, we can use both in any situation but there is 1 point to pay attention which that if we use empty square brackets then we need to initialized the string while we are declaring it, means we need to assigned value to string we define it.

char str [ ] = “VANSHIT GARG”; (✔)

char str[ ]; (❌)

str[ ] = “VANSHIT GARG”;

char str[20] = “VANSHIT GARG”; (✔)

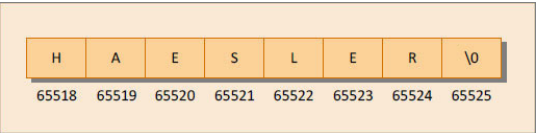
In this case also to put string after declaration we need to use some special method we can’t enter like

char str[20];

str[20]=”Vanshit Garg”;

What Happens in Memory?

The elements of the string are stored in contiguous memory locations.



String Format Specifier?

String I/O?

**scanf()** is not capable of receiving multi-word strings like “Vanshit Garg”. So we use **gets() & puts()** functions.

**# include <stdio.h>**

**int main()**

**{**

**char name[25];**

**printf ("Enter your full name: ");**

**gets (name);**

**puts (“Hello!”);**

**puts (name);**

**return 0;**

**}**

**Output: Enter your full name: Vanshit Garg**

**Hello!**

**Vanshit Garg**

As in this program we can see than gets()issue for taking a string at run time and puts() is use to display the string with we not need to make a loop to print a string, but puts() can display only one string at a time and it print the string in new line automatically. Similarly gets() also take only 1 string at a time.

But there is a way by which we can take string by using scanf() and even we can take multiple strings.

**char name[25];**

**printf (“Enter your full name ”);**

**scanf (“%[^\n]s”, name);**

In this case scanf() will keep on receiving characters in name until it gets \n, but this is not the best way to do this.

**Note :** gets() is dangerous & outdated so we can use fgets(str, n, file) function to get string at run time, it stops when n-1 characters are inputted or new line is entered.

String Functions?

String has a library which has function related to it and to use these functions we need to include the library name in the program which **<string.h>** & below is the table containing the string related functions.

It is also know as **“Standard Library Function for String”**.

|  |  |
| --- | --- |
| **Function** | **Use** |
| strlen | Finds length of a string |
| strlwr | Converts a string to lowercase |
| strupr | Converts a string to uppercase |
| strcat | Appends one string at the end of another |
| strncat | Appends first n characters of a string at the end of another |
| strcpy | Copies a string into another |
| strncpy | Copies first n characters of one string into another |
| strcmp | Compares two strings |
| strncmp | Compares first n characters of two strings |
| strcmpi | Compares two strings by ignoring the case |
| stricmp | Compares two strings without regard to case (identical to strempi) |
| strnicmp | Compares first n characters of two strings without regard to case |
| strdup | Duplicates a string |
| strehr | Finds first occurrence of a given character in a string |
| sterchr | Finds last occurrence of a given character in a string |
| strstr | Finds first occurrence of a given character in a string |
| strset | Sets all characters of string to a given character |
| strnset | Sets first n characters of a string to a given character |
| strrev | Reverses string |

Role of Pointer in String?

11

Structures

* What is Structures?
* Syntax of Structures?
* Memory allocation for Structures?
* Benefits of using Structures?
* Array of Structures?
* Initializing of Structures?
* Pointers in Structures?
* What is Arrow Operator & it’s use?
* Passing Structure to Function?
* What is Typedef keyword?

**Objectives**

What is Structures?

Syntax of Structures?

Memory allocation for Structures?

Benefits of using Structures?

Array of Structures?

Initializing of Structures?

Pointers in Structures?

What is Arrow Operator & it’s use?

Passing Structure to Function?

What is Typedef keyword?

Typedef – It allows is to change name of a keyword.

#include <stdio.h>

int main()

{

    typedef int vansh;

    vansh num = 45;

    int num2 = 55;

    printf("Value of num is %d \n",num);

    printf("Value of num2 is %d \n",num2);

    return 0;

}

As here “int” keyword is change to “vansh” means now we can write vansh in place of int, but it not means that we can use int.

12

File I/O

* What is File I/O?
* Operation for Files & Types of Files?
* Role of Pointer in File Handling?
* Opening & Closing of a File?
* File opening Modes?
* Reading & Writing of a file?
* Read & Write a Character?
* What is End of File?

**Objectives**

What is File I/O?

Operation for Files & Types of Files?

Role of Pointer in File Handling?

Opening & Closing of a File?

File opening Modes?

Reading & Writing of a file?

Read & Write a Character?

What is End of File?

13

Dynamic Memory Allocation

* What is Dynamic Memory Allocation?
* Function for DMA?

**Objectives**

What is Dynamic Memory Allocation?

Function for DMA?

//// EXTRA

Extra Commands

Both these command work in “Turbo C” not in “C”.

clrscr (); - allows us to clear the clean/terminal. It must be written after declaration of variables otherwise compiler show an error.

getch (); - waits for the user to input data.

#include <stdio.h>

#include <conio.h>

int

main ()

{

  int a = 45;

  clrscr ();

  printf ("value of a is %d", a);

  return 0;

  getch ();

}